

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-006805

(43)Date of publication of application : 12.01.1996

(51)Int.Cl.

G06F 9/46

G06F 15/16

(21)Application number : 06-132878

(71)Applicant : TOSHIBA CORP

(22)Date of filing : 15.06.1994

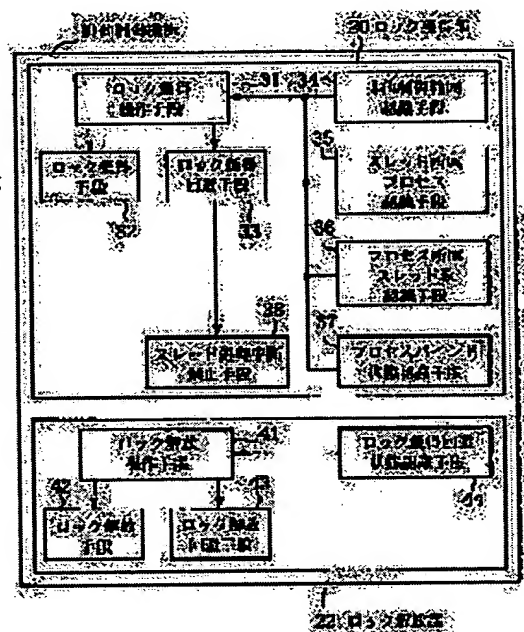
(72)Inventor : YASUI KEISUKE

## (54) MULTIPROCESSOR SYSTEM

## (57)Abstract:

PURPOSE: To effectively utilize respectively processors under optimum lock control.

CONSTITUTION: This multiprocessor system is equipped with an exclusive control range recognizing means 34 which recognizes whether or not data accessed by a thread are data shared only by threads in the same process, a thread belonging process recognizing means 35 which recognizes the process that the thread being executed belongs to, a process belonging thread quantity recognizing means 36 which recognizes the number of threads belonging to the process that the thread being executes belongs to, and a lock acquiring operation means 31 which evades lock acquisition when recognizing that the process that the thread being executed belongs has only the thread in a critical access where the data shared only by threads in the same process are accessed on the basis of the recognition results of the respective recognizing means 34, 35, and 36.



## LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平8-6805

(43) 公開日 平成8年(1996)1月12日

(51) Int.Cl.<sup>8</sup>

G 0 6 F 9/46  
15/16

識別記号

3 6 0 D 7737-5B  
3 4 0 Z

庁内整理番号

F I

技術表示箇所

審査請求 未請求 請求項の数 3 O L (全 7 頁)

(21) 出願番号 特願平6-132878

(22) 出願日 平成6年(1994)6月15日

(71) 出願人 000003078

株式会社東芝

神奈川県川崎市幸区堀川町72番地

(72) 発明者 安井 啓介

東京都青梅市末広町2丁目9番地 株式会

社東芝青梅工場内

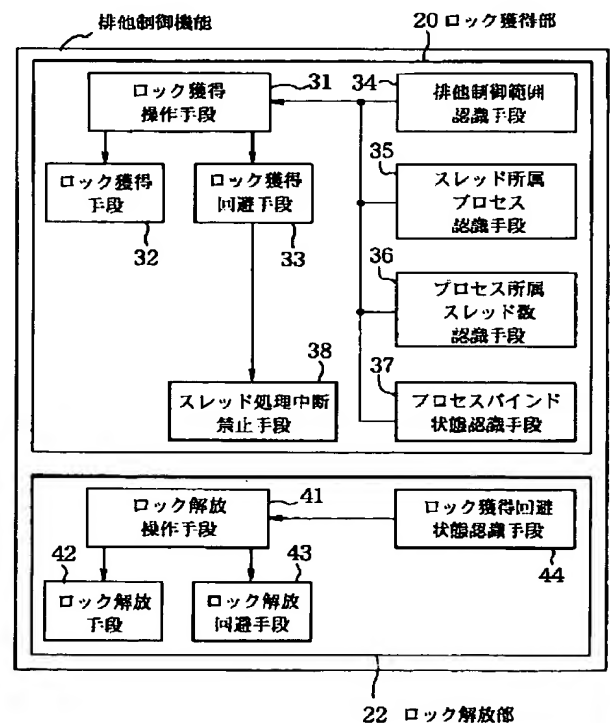
(74) 代理人 弁理士 鈴江 武彦

(54) 【発明の名称】 マルチプロセッサシステム

(57) 【要約】

【目的】最適なロック制御によって各プロセッサを有効利用することを可能にする。

【構成】スレッドによってアクセスされるデータが、同一プロセス内のスレッドのみによって共有されるデータであるか否かを認識する排他制御範囲認識手段34と、実行中のスレッドが属するプロセスを認識するスレッド所属プロセス認識手段35と、実行中のスレッドが属するプロセスに所属するスレッドの数を認識するプロセス所属スレッド数認識手段36と、各認識手段34、35、36による認識結果に基づいて、同一プロセス内のスレッドのみによって共有されるデータへアクセスするクリティカル領域で、実行中のスレッドが属するプロセスが、そのスレッド1つしか持たないことを認識した場合には、ロック獲得を回避させるロック獲得操作手段31を具備して構成する。



## 【特許請求の範囲】

【請求項 1】 1つのプロセス中に複数のスレッドを定義することが可能なマルチスレッド構造のオペレーティングシステムに基づいて動作するものであって、キャッシュを装備する複数のプロセッサ上で動作するプロセスによってデータが共有されるマルチプロセッサシステムにおいて、

スレッドによってアクセスされるデータが、同一プロセス内のスレッドのみによって共有されるデータであるかを認識する排他制御範囲認識手段と、

実行中のスレッドが属するプロセスを認識するスレッド所属プロセス認識手段と、

前記実行中のスレッドが属するプロセスに所属するスレッド数を認識するプロセス所属スレッド数認識手段と、前記排他制御範囲認識手段、前記スレッド所属プロセス認識手段、及び前記プロセス所属スレッド数認識手段による認識結果に基づいて、同一プロセス内のスレッドのみによって共有されるデータへアクセスするクリティカル領域で、前記実行中のスレッドが属するプロセスが、そのスレッド 1 つしか持たないことを認識した場合には、ロック獲得を回避させるロック獲得操作手段と、を具備したことを特徴とするマルチプロセッサシステム。

【請求項 2】 同一プロセス内のスレッドのみによって共有されるデータへアクセスする時、前記実行中のスレッドが属するプロセスの持つ全てのスレッドが同じプロセスにバインドされているかを認識するプロセスバインド状態認識手段をさらに具備し、前記ロック獲得操作手段は、前記プロセスバインド状態認識手段による認識結果に基づいて、クリティカル領域の処理の間にスレッドの処理実行が中断されないことを保証した上で、ロック獲得を回避させることを特徴とする請求項 1 記載のマルチプロセッサシステム。

【請求項 3】 前記クリティカル領域の処理が終了する際に、前記ロック獲得操作手段によってロック獲得が回避されている場合には、ロック解放を回避させるロック解放操作手段をさらに具備したことを特徴とする請求項 1 または請求項 2 記載のマルチプロセッサシステム。

## 【発明の詳細な説明】

## 【0001】

【産業上の利用分野】 本発明は、マルチスレッド構造のオペレーティングシステムによって動作するマルチプロセッサシステム、及び同システムにおける共有メモリの排他制御方法に関する。

## 【0002】

【従来の技術】 一般に、共有メモリを有し、各プロセッサにキャッシュが設けられているマルチプロセッサプロセッサシステムでは、複数のプロセッサによって共有される共有データにアクセスするクリティカル領域では、ロック操作などを行ない、権利を与えられたプロセスだ

2

けが、対象とするデータにアクセスできるようにする排他制御が行なわれている。マルチプロセッサシステムが、マルチスレッド構造のオペレーティングシステムによって動作するのであれば、スレッドに対してアクセスの権利が与えられる。

【0003】 また、この種、排他制御では、共有データが変更されると、変更されたことが他のプロセッサに通知される。共有データの変更が通知されたプロセッサは、同プロセッサが持つキャッシュに同じデータが保持されていると、そのデータを無効化あるいは更新して、データの一貫性がとられるようにしている。

【0004】 一般に使用される排他制御の方法としてロック操作がある。ロック操作では、ある共有データに対して共有メモリ上にロック変数を用意し、共有データを処理する前にロック変数を確保（ロック変数にフラグを立てたり、所有者名を格納する）し、共有データをアクセスした後でロック変数をクリアする。

【0005】 プロセス（またはスレッド）は、ロック変数を確保しようとする際に、既に他のプロセス（またはスレッド）が、そのロック変数を確保している場合には、そのロック変数がクリアされるまで待ち状態となり、クリアされると直ちにロック変数を確保する。これによって、2つのプロセッサが同時に同じデータにアクセスして、データが不整合になることを防ぐことができる。

【0006】 ロック操作というのはマルチプロセッサの間での排他制御を行なうため、各プロセッサが持つキャッシュ上のデータについて更新等を行なうのみでなく、共有メモリ上に確保されたロック変数等のデータの更新を行なわなければならない。

【0007】 しかし、マルチプロセッサシステムにおいて、共有メモリ上のデータの更新に要する時間は、キャッシュ上のデータの更新に比べて非常に多く要する。従って、ロック操作を少なくすることができれば処理時間が短縮され、プロセッサを有効利用することができる。

【0008】 特にマルチスレッド構造のオペレーティングシステムによって動作するマルチプロセッサシステムでは、同一プロセス内のスレッドのみによって共有されるデータにアクセスする時に、そのスレッドが属するプロセスが持つスレッドが自分のみである場合のロック操作のような、不要なロック操作を避けることがプロセッサの有効利用につながる。

## 【0009】

【発明が解決しようとする課題】 このように従来のマルチスレッド構造のオペレーティングシステムによって動作するマルチプロセッサシステムにおいては、必要以上のロック操作が存在しているため、プロセッサが有効利用されないという問題があった。

【0010】 本発明は前記のような事情を考慮してなされたもので、最適なロック制御によって各プロセッサを

有効利用することが可能なマルチプロセッサシステムを提供することを目的とする。

#### 【0011】

【課題を解決するための手段】本発明は、1つのプロセス中に複数のスレッドを定義することが可能なマルチスレッド構造のオペレーティングシステムに基づいて動作するものであって、キャッシュを装備する複数のプロセッサ上で動作するプロセスによってデータが共有されるマルチプロセッサシステムにおいて、スレッドによってアクセスされるデータが、同一プロセス内のスレッドのみによって共有されるデータであるか否かを認識する排他制御範囲認識手段と、実行中のスレッドが属するプロセスを認識するスレッド所属プロセス認識手段と、前記実行中のスレッドが属するプロセスに所属するスレッドの数を認識するプロセス所属スレッド数認識手段と、前記排他制御範囲認識手段、前記スレッド所属プロセス認識手段、及び前記プロセス所属スレッド数認識手段による認識結果に基づいて、同一プロセス内のスレッドのみによって共有されるデータへアクセスするクリティカル領域で、前記実行中のスレッドが属するプロセスが、そのスレッド1つしか持たないことを認識した場合には、ロック獲得を回避させるロック獲得操作手段とを具備したことを特徴とする。

【0012】また、同一プロセス内のスレッドのみによって共有されるデータへアクセスする時、前記実行中のスレッドが属するプロセスの持つ全てのスレッドが同じプロセッサにバインドされているかを認識するプロセスバインド状態認識手段をさらに具備し、前記ロック獲得操作手段は、前記プロセスバインド状態認識手段による認識結果に基づいて、クリティカル領域の処理の間にスレッドの処理実行が中断されないことを保証した上で、ロック獲得を回避させることを特徴とする。

【0013】また、前記クリティカル領域の処理が終了する際に、前記ロック獲得操作手段によってロック獲得が回避されている場合には、ロック解放を回避させるロック解放操作手段をさらに具備したことを特徴とする。

#### 【0014】

【作用】このような構成によれば、共有データにアクセスするクリティカル領域を排他制御する際に、プロセスとスレッド数の関係、プロセッサとプロセスのバインド状態を考慮して、ロック操作を行なうことによって、プロセッサを有効利用することができる。

【0015】つまり、同一プロセス内のスレッドのみによって共有されるデータへアクセスする時に、実行中のスレッドが属するプロセス中に含まれるスレッド数が1つである場合、また、同一プロセス内のスレッドのみによって共有されるデータへアクセスする時に、実行中のスレッドが属するプロセスの持つ全てのスレッドが同じプロセッサにバインドされている場合には、共有データに対してロックを獲得しなくても問題ないものとしてロ

ック獲得処理を省略する。

#### 【0016】

【実施例】以下、図面を参照して本発明の一実施例を説明する。図1は本発明の一実施例に係わるマルチプロセッサシステムの概略構成を示すブロック図である。図1に示すように、マルチプロセッサシステムは、複数のプロセッサ10-1, 10-2, ..., 10-nが設けられている。各プロセッサ10-1, 10-2, ..., 10-nは、それぞれキャッシュ12-1, 12-2, ..., 12-nを装備している。プロセッサ10-1, 10-2, ..., 10-nは、バス14を介して共有メモリ16と接続される。

【0017】共有メモリ16は、各プロセッサ10-1, 10-2, ..., 10-nが動作するためのマルチスレッド構造のオペレーティングシステムその他、各種データを格納している。また、共有メモリ16には、各プロセッサ10-1, 10-2, ..., 10-nが、共有データをアクセスする際のロック制御に用いられるロック変数が格納される。

【0018】各プロセッサ10-1, 10-2, ..., 10-nは、共有メモリ16に格納されたプログラムに基づいて実現されるものであって、共有メモリ16に格納されたデータに対する排他制御を行なう排他制御機能18-1, 18-2, ..., 18-nを有している。共有メモリ16には、排他制御機能18-1, 18-2, ..., 18-nによって参照されるロック制御情報19（詳細については後述する）が格納されている。

【0019】排他制御機能18-1, 18-2, ..., 18-nは、同一プロセス内のスレッドのみによって共有されるデータへアクセスするクリティカル領域において、ロック操作を行なわなくても障害等が発生しないような状況にはロック操作を回避するように機能するものである。

【0020】図2は本発明による排他制御方法を実現する排他制御機能18-1, 18-2, ..., 18-nの構成を示す機能ブロック図である。図2に示すように、本発明の排他制御機能18-1, 18-2, ..., 18-nは、ロック獲得操作を行なうロック獲得部20、及びロック解放操作を行なうロック解放部22から構成される。

【0021】さらに、ロック獲得部20は、ロック獲得操作手段31、ロック獲得手段32、ロック獲得回避手段33、排他制御範囲認識手段34、スレッド所属プロセス認識手段35、プロセス所属スレッド数認識手段36、プロセスバインド状態認識手段37、及びスレッド処理中断禁止手段38によって構成されている。

【0022】また、ロック解放部22は、ロック解放操作手段41、ロック解放手段42、ロック解放回避手段43、及びロック獲得回避状態認識手段44によって構成されている。

【0023】ロック獲得操作手段31は、排他制御範囲認識手段34、スレッド所属プロセス認識手段35、プロセス所属スレッド数認識手段36、及びプロセスバインド状態認識手段37による各認識結果に基づいて、ロック獲得手段32とロック獲得回避手段33を使い分けて、クリティカル領域を排他制御するためのロック獲得処理を実行する。すなわち、ロック獲得操作手段31は、プロセスとスレッド数の関係や、プロセッサとプロセスのバインド状態を考慮したロック操作を行なう。

【0024】ロック獲得手段32は、共有データをアクセスする際に、ロック獲得操作手段31によってロック獲得すべきものとされた場合に、ロックを獲得するものである。

【0025】ロック獲得回避手段33は、共有データをアクセスする際に、ロック獲得操作手段31によってロック獲得の必要がないとされた場合にロック獲得回避処理を行なう。また、ロック獲得回避手段33は、ロック獲得を回避する際に、スレッド処理中断禁止手段38によって他のスレッドへ実行権を切り替えることを禁止させる。

【0026】排他制御範囲認識手段34は、共有データが、同じプロセス中のスレッドのみによって共有（アクセス）されるのか（プロセスローカル）、あるいは異なるプロセス中のスレッドでアクセスされるのか（プロセスグローバル）を、ロック制御情報19のロック構造体（後述する）を参照して認識するものである。

【0027】スレッド所属プロセス認識手段35は、実行中のスレッドが所属するプロセスを、ロック制御情報19のスレッド構造体（後述する）を参照して認識するものである。

【0028】プロセス所属スレッド数認識手段36は、実行中のスレッドが属するプロセスに所属するスレッドの数を、ロック制御情報19のプロセス構造体（後述する）を参照して認識するものである。

【0029】プロセスバインド状態認識手段37は、実行中のスレッドが属するプロセスの持つ全てのスレッドが同じプロセッサにバインドされているかを、ロック制御情報19のプロセス構造体（後述する）を参照して認識するものである。

【0030】スレッド処理中断禁止手段38は、ロック獲得を回避する場合に、他のスレッドへ実行権を切り替えることを禁止する指示を行なうものである。スレッド処理中断禁止手段38は、システム側からの処理の中断要求に対しても、これを拒否する。

【0031】ロック解放操作手段41は、ロック獲得回避状態認識手段44による認識結果に基づいて、ロック解放手段42、及びロック解放回避手段43を使い分けて、ロック獲得部20によるロック獲得操作に対応するロック解放操作を行なうものである。

【0032】ロック解放手段42は、ロック獲得部20

によってロックが獲得されている場合に、クリティカル領域の処理が終了した後に、ロック解放処理を行なう。ロック解放回避手段43は、ロック獲得部20によってロック獲得が回避されている場合に、ロック解放回避処理を行なう。

【0033】ロック獲得回避状態認識手段44は、ロック制御情報19のプロセッサ構造体（後述する）を参照して、ロック制御部20によってロックの獲得の回避がなされているかを認識するものである。

【0034】次に、共有メモリ16に格納されるロック制御情報19について説明する。ロック制御情報19は、共有メモリ16中に存在するが、排他制御機能18-1, 18-2, ..., 18-nによって参照される際には、対応するプロセッサ10-1, 10-2, ..., 10-nのキャッシュ12-1, 12-2, ..., 12-nにも格納される。

【0035】ロック制御情報19は、図3に示すように、ロック構造体50、プロセス構造体51、スレッド構造体52、及びプロセッサ構造体53に関する情報が格納されている。

【0036】ロックのためのデータが格納されたロック構造体50には、排他制御範囲26に関する情報が含まれている。排他制御範囲26は、ロックが保護する共有データが、あるプロセス内のスレッドのみにアクセスされるか、すなわち同じプロセス中のスレッド間で共有データにアクセスするプロセスローカルであるか、異なるプロセス中のスレッド間で共有データにアクセスするプロセスグローバルであるかを示す情報が設定されている。排他制御範囲26に関する情報は、ロック構造体50の初期化時に設定される。各スレッドが扱うデータは、対象が決まっているので、各スレッドの関係（プロセスグローバル、プロセスローカル）が初期化時に予め設定される。

【0037】プロセスに関するデータが格納されているプロセス構造体51には、プロセスのスレッド数55及びバインド状態56に関する情報が含まれている。スレッド数55は、プロセスが持つスレッド数であり、バインド状態29は、プロセスのプロセッサへのバインド状態を示している。

【0038】スレッドに関するデータが格納されているスレッド構造体52には、スレッドの所属プロセス57を示す情報が含まれている。プロセッサに関するデータが格納されるプロセッサ構造体53には、プロセッサに対するロック回避状態58及び実行中断禁止状態59に関する情報が含まれている。ロック回避状態58は、プロセッサが、ロック獲得を回避している状態であるか否かを示し、実行中断禁止状態59は、ロック操作を回避している場合にクリティカル領域の処理中には他のスレッドに処理が切り替わらないように、スレッドの処理の中断を制限することを示している。

【0039】次に、本実施例の動作について説明する。まず、ロック獲得操作の処理について、図4に示すフローチャートを参照しながら説明する。まず、共有メモリ16に格納された共有データに対してアクセスするためにロック獲得操作を実行する際、ロック獲得部20の排他制御範囲認識手段34は、ロック制御情報19のロック構造体50の排他制御範囲54の情報を参照して、排他制御範囲の確認処理を行なう（ステップA1）。すなわち、プロセスグローバルであるかプロセスローカルであるかを認識する。

【0040】ここで、共有データがローカルデータであって、同共有データにアクセスしようとするスレッドを含むプロセス中の他のスレッドのみにアクセスされると認識された場合には、スレッド所属プロセス認識手段35は、実行中のスレッドが属するプロセスの持つスレッド数を、プロセス構造体51のスレッド数55を参照して認識する（ステップA2）。

【0041】この結果、プロセスの持つスレッド数が複数であった場合、プロセスバインド状態認識手段37は、実行中のスレッドが属するプロセスのバインド状態を、プロセス構造体51のバインド状態を参照して認識する（ステップA3）。

【0042】プロセスが実行中のプロセッサにバインドされていない場合には、ロック獲得操作手段31は、ロック獲得手段32を用いて、すぐにロック獲得処理を行なう（ステップA4）。

【0043】また、プロセスが実行中のプロセッサにバインドされている場合、すなわちプロセス中のプロセスローカルの関係にある全てのスレッドが同じプロセッサにバインドされている場合には、ロック獲得操作手段31は、ロックの必要がないものとして、ロック獲得回避手段33を用いて実行中断禁止処理を実行する。すなわち、スレッド処理中断禁止手段38によって、プロセッサ構造体53の実行中断禁止状態59に実行中断禁止を示す情報をセットし、他のスレッドへ実行権を切り替えることを禁止する。

【0044】他のスレッドへ実行権を切り替える実行中断禁止処理は、例えばTSS（タイムシェアリングシステム）等のために、強制的にシステムによって処理が中断されるような状況であっても、排他制御機能（処理中断禁止手段38）のレベルで実行中断を禁止する。

【0045】次に、ロック獲得操作手段31は、ロック獲得回避手段33を用いて、ロック獲得回避処理を実行する（ステップA6）。すなわち、プロセッサ構造体53のロック回避状態58に、ロック獲得回避を示す情報をセットする。

【0046】なお、ステップA2において、スレッド所属プロセス認識手段35が、実行中のスレッドが属するプロセスの持つスレッド数を、プロセス構造体51のスレッド数55を参照して認識した結果、プロセス中に1

つのスレッドしかない場合には、ロック獲得操作手段31は、ロックの必要がないものとして、ロック獲得回避手段33を用いて、ロック獲得回避処理を実行する（ステップA6）。すなわち、プロセッサ構造体53のロック回避状態58に、ロック獲得回避を示す情報をセットする。

【0047】一方、ステップA1において、ロックが保護する共有データが、他のプロセスからもアクセスされるグローバルデータであると認識された場合には、ロック獲得操作手段31は、ロック獲得手段32を用いて、すぐにロック獲得処理を行なう（ステップA4）。

【0048】こうして、ステップA4におけるロック獲得処理、またはステップA6におけるロック獲得回避処理（ロックを獲得せずに）が完了すると、クリティカル領域の処理に移る。

【0049】ロック獲得部20における処理では、共有メモリ16に格納された共有データにアクセスする際に、ロック獲得を行なわなくても問題がない場合にはロック獲得処理を省いている。また、クリティカル領域の排他制御を行なう際には、ロック制御情報19がキャッシュ上に存在しているので、実行中断禁止処理、ロック獲得回避処理は、高速に実行することができる。

【0050】次に、ロック解放操作の処理について、図5に示すフローチャートを参照しながら説明する。まず、クリティカル領域の処理が終了したところで、ロック解放操作を実行する際、ロック解放部22のロック獲得回避状態認識手段44は、ロック制御情報19のプロセッサ構造体53のロック回避状態58の情報を参照して、ロック回避状態の確認処理を行なう（ステップB1）。すなわち、ロック回避状態58に、ロック獲得回避を示す情報がセットされているか否かを認識する。

【0051】ここで、ロック獲得状態にあると認識された場合には、ロック解放操作手段41は、ロック解放手段42を用いてロック解放処理を行ない、排他制御処理を終了する（ステップB4）。

【0052】また、ロック回避状態にあると認識された場合には、ロック解放操作手段41は、ロック解放回避手段43を用いて、ロック回避状態解除処理を行なう。すなわち、ロック解放回避手段43は、ロック制御情報19のプロセッサ構造体53のロック回避状態33をリセットする（ステップB2）。

【0053】さらに、ロック解放操作手段41は、ロック解放回避手段43を用いて、実行中断禁止解除処理を行なう。すなわち、ロック解放回避手段43は、ロック制御情報19のプロセッサ構造体53の実行中断禁止状態59をリセットし、排他制御処理を終了する（ステップB3）。すなわち、共有データに対するロック制御を回避している場合には、クリティカル領域の処理を出る時には、ロックの解放処理を行なわない。

【0054】なお、前記実施例では、単純なクリティカ

9

ル領域に対する排他制御について説明しているが、例えばロックが階層的に行なわれるような処理については次のようにすることもできる。

【0055】ロック獲得が必要となった場合、ロック獲得操作手段31は、ロック獲得手段32を用いて、ロック獲得処理を実行する。この際、ロック制御情報19のプロセッサ構造体53の実行中断禁止状態を、ロックの階層が上がる毎に順次、階層のレベルに応じて更新する。

【0056】そして、各階層レベルのクリティカル領域の処理が終了する毎に、ロック解放操作手段41は、ロック解放手段42を用いて、階層毎に順次ロックを解放していく。

【0057】また、ロック獲得回避処理が階層的にされている場合には、同様にしてロック回避状態解除処理を、階層毎に順次行なう。このようにして、同一プロセス内のスレッドのみによって共有されるデータへアクセスするクリティカル領域で、前記実行中のスレッドが属するプロセスが、そのスレッド1つしか持たない場合、あるいは実行中のスレッドが属するプロセスの持つ全てのスレッドが同じプロセッサにバインドされている場合には、ロック操作を行なわないことにより、プロセッサを有効に利用することができる。

【0058】特に、共有メモリ16に対する書き込み（ロック変数に対する）に時間がかかったり、ロック操作に伴い他のプロセッサに対する通知等に多くのコストを要するシステムにおいては、ロック操作を省くことで多くの効果が得られる。

【0059】

【発明の効果】以上詳述したように本発明によれば、プ

10

ロセスとスレッド数の関係、プロセッサとプロセスのバインド状態を考慮して、不要なロック制御を省くことによって、最適なロック制御によって各プロセッサを有効利用することが可能となるものである。

【図面の簡単な説明】

【図1】本発明の一実施例に係わるマルチプロセッサシステムの概略構成を示すブロック図。

【図2】本発明による排他制御方法を実現する排他制御機能18-1、18-2、…、18-nの構成を示す機能ブロック図。

【図3】本実施例におけるロック制御情報19に含まれる各種情報の一例を示す図。

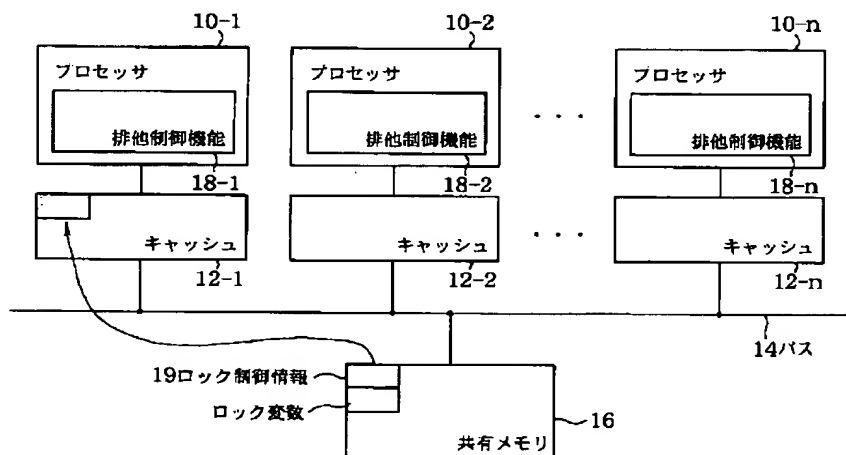
【図4】本実施例におけるロック獲得操作の処理を説明するためのフローチャート。

【図5】本実施例におけるロック解放操作の処理を説明するためのフローチャート。

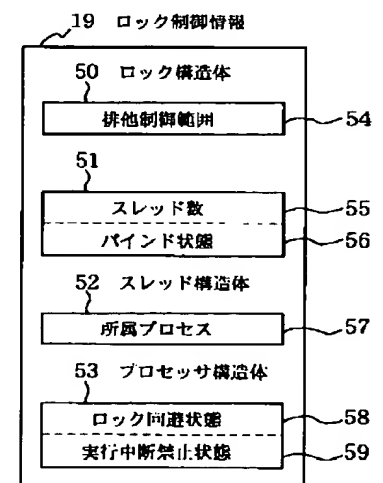
【符号の説明】

10-1、10-2、…、10-n…プロセッサ、12-1、12-2、…、12-n…キャッシュ、14…バス、16…共有メモリ、18-1、18-2、…、18-n…排他制御機能、19…ロック制御情報、20…ロック獲得部、22…ロック解放部、31…ロック獲得操作手段、32…ロック獲得手段、33…ロック獲得回避手段、34…排他制御範囲認識手段、35…スレッド所属プロセス認識手段、36…プロセス所属スレッド数認識手段、37…プロセスバインド状態認識手段、38…スレッド処理中断禁止手段、41…ロック解放操作手段、42…ロック解放手段、43…ロック解放回避手段、44…ロック獲得回避状態認識手段。

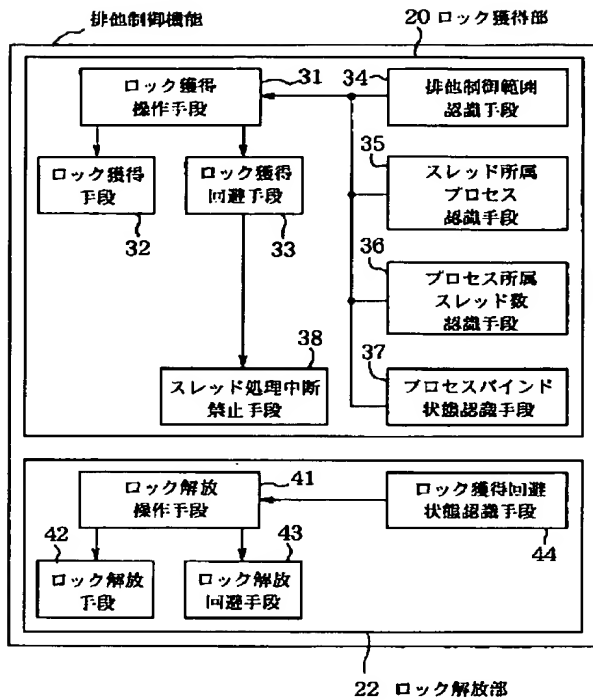
【図1】



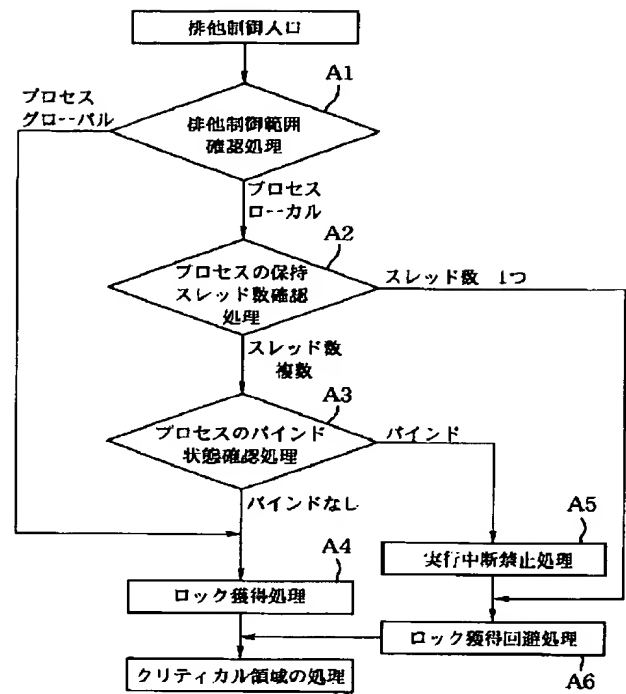
【図3】



【図 2】



【図 4】



【図 5】

